Workflow Management

Software Requirements Specification

Submitted in partial fulfillment of the requirements of the course CS223 – Software Engineering

Submitted by

Chitraksh Sadayat (B16CS007)

Vishakh Suresh (B16CS038)

Revision History

Date	Version	Description
09 February 2018	1.0	First draft
02 March 2018	2.0	Second draft

Table of Contents

Software Requirements Specification

1. Introduction

The introduction provides an overview of the software requirements specifications. It includes the purpose, scope, constraints, organization, assumptions, abbreviations and references pertaining to this document.

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for a Workflow management system in an academic institute. The intended audience to use this software are the students, research scholars, faculty, the Director and the Office of Academics of our institute.

1.2 Scope

The purpose of this Workflow Management System (WMS) is to create a convenient and easy-to-use environment for the users who are trying to apply for a leave or financial assistance. The system will process the requests following a specific hierarchy, which is described in the Assumptions and Dependencies section below. In addition to applying for a leave or financial assistance, a user is allowed to check the status of his leave or financial assistance request and also to approve the requests of those users who are under him/her in the hierarchy pyramid.

1.3 Constraints

- There are no hardware constraints.
- Software constraints
  - Development of the software has to be done exclusively in C++.
  - Concurrent updates cannot be permitted as the program runs.
- Development constraints: The software has to be developed using 2 developers.

1.4 Assumptions and Dependencies

The following hierarchy is assumed for leave and financial assistance approval.

- An undergraduate's leave request is processed directly by the Administrator .i.e. Undergrads → Administrator.
- A research scholar's leave request is processed by his faculty advisor before it is sent to the office of academics for its approval.
  .i.e. Research Scholar → Faculty (Supervisor) → Administrator.
- A faculty's leave request is processed following the hierarchy below: Faculty → Head of the Department → Director → Administrator.

- A H.O.D's request is processed through the Director before it is sent to the Administrator for implementation.
  .i.e. Head of a Department → Director → Administrator.
- The Director may directly send his leave application to the Administrator for implementation.
- The Students (Undergraduates and Research scholars) may apply for financial assistance.
- Their applications are processed following the same hierarchy as described above.

1.5 Definitions, Acronyms and Abbreviations

Term	Definition
Leave	The leave object keeps track of remaining leaves. It keeps track of the status while processing leave applications. It includes the date of issue and who has issued it.
Finance	A finance object is used for applying financial assistance application. It contains family income of applicant for comparison based approval and can be used to track status of approval like leave.
User	Any individual in the academic institute.
Student	One who studies in the academic institute
Employee	One who is employed in the academic institute
Undergrads	A category of student class
Research _Scholar	A category of student class studying under a faculty
Faculty	An employee teaching in the institute and assisting a research scholar
HOD	He is the head of the department and an employee in the institute
Director	He is an employee and is the head of the academic institute
Administrator	A passive actor. It refers to the office of academics whose final approval is required for approving any application.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the various constraints under which it operates. For example, this document.
Stakeholder	Any person other than the developer, with an interest in the project.

1.6 References

[1] https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

[2] www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf
[3] https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc
[4] www2.latech.edu/~box/ase/srs_template.doc
[5] web.itu.edu.tr/~tanriverdin/proje/SRSSample.doc
[6] https://www.onedesk.com/writing-a-software-requirements-specification-document/
[7] https://www.geeksforgeeks.org/how-to-write-a-good-srs-for-your-project
[8] https://en.wikipedia.org/wiki/Software_requirements_specification

1.7 Organization of the Document

The SRS document is basically organised into four major sections.

1. The first section is an introductory overview of the SRS and lays foundational information regarding the software implemented. It highlights the basic purpose, scope and expected readership. It defines the general terms of frequent usage and highlights the concrete assumptions used in the software implementation. Finally, it obliges the support sources for making the SRS development possible by acknowledging the reference.
2. The second section describes all the product functionalities categorised into functional and non functional requirements. The implementation and design is clarified using the UML models and diagrams. The user characteristics provide a detailed vision about the end users and customers.
3. The third section highlights the system characteristics such as performance, reliability and constraints involved in the development process. A detailed use case description is provided for clarity and comprehensibility of the use cases implemented.
4. The final segment incorporates all the supporting information in the form of information rich and descriptive appendices required by all the readers of the SRS document for better understanding of the software.

2. Overall Description

2.1 Product Functions

2.1.1 Functional Requirements

2.1.1.1 Apply Leave

- **Introduction**
  Any user in the academic institute can apply for leave.
- **Inputs**
  There are 2 inputs: the first one being the number of leaves he/she wants to apply and the second one being the Leave details which include the number of permittable leaves, date of issue and the designation of the user.
- **Processing**
  If the number of leaves the user has applied for, exceeds the number of permittable leaves, the application is rejected, else it is pemitted.
- **Output**
  If the leave is permittable, then it is forwarded to the next higher authority for approval and he gets a notification. If the leave is not permittable, the applicant gets a notification, asking him to reenter the number of days.

2.1.1.2 Apply Finance

- **Introduction**
  Any student in the academic institute can apply for financial assistance based on his/her family income.
- **Inputs**
  There are 2 inputs: the first one being amount he/she wants to apply for and the second one being the financial details which include the designation of the user and his/her family income.
- **Processing**
  The maximum permittable amount is determined by comparison of the applicant's family income with a set of standard benchmarks. The application is accepted only if the amount requested for is less than the maximum permitable amount.
- **Output**
  If the application is accepted, then it is forwarded to the next higher authority for approval and he gets a notification. If not, the applicant gets a notification, asking him to reenter the amount.

2.1.1.3 Approve Leave

- **Introduction**

  The user may approve the leave applications which are awaiting his/her approval and forward to the next higher authority.
- **Inputs**

  The Leave details which include the number of permittable leaves, the current status of the application, date of issue and the designation of the user.
- **Processing**

  If any non-administrative user approves the leave requests that have reached him through the hierarchy chain, it is forwarded to the next higher authority. If the Administrator approves the leave, the Leave is granted and the maximum permitted leaves are updated accordingly.
- **Output**

  There is no output to this function as it simply performs leave status and permitted leave updation.

2.1.1.4 Approve Finance

- **Introduction**

  The user may approve the financial assistance applications which are awaiting his/her approval and forward to the next higher authority.
- **Inputs**

  The financial details which include the designation of the user, his/her family income, the status of the application and the amount he is currently receiving.
- **Processing**

  If any non-administrative user approves the financial assistance requests that have reached him through the hierarchy chain, it is forwarded to the next higher authority. If the Administrator approves the application, financial assistance is granted and the amount which he receives as assistance is updated accordingly.
- **Output**

  There is no output to this function as it simply performs financial assistance status and amount updation.

2.1.1.5 Check Leave Status

- **Introduction**

  The leave applicant can track his application.

- **Inputs**

  There is no input to this function.

- **Output**

  There is no processing involved. The function just outputs the current status of the application.

2.1.1.6 Check Financial Assistance Status

- **Introduction**

  The user who has applied for financial assistance can track his application.

- **Inputs**

  There is no input to this function.

- **Output**

  There is no processing involved. The function just outputs the current status of the application.

2.1.2 Non-functional requirements

2.1.2.1 Performance Requirements

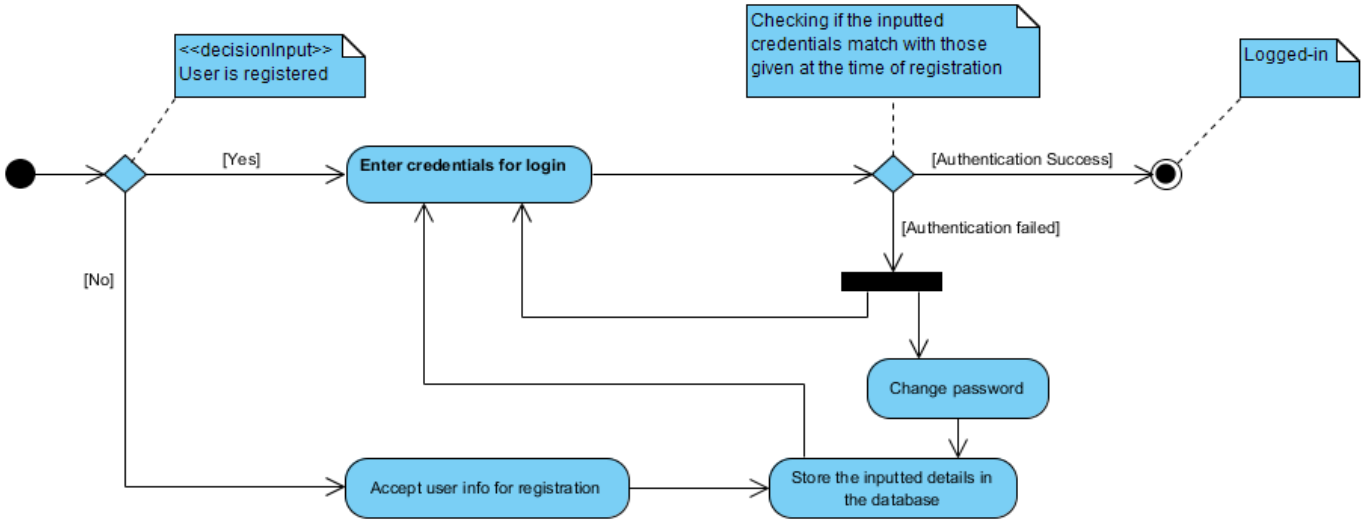
Performance may not be a big issue. The main task is concerned with checking status and status queries involve small pieces of data because we already have the leave/financial status as an object inside our user object on which we are operating. So time taken is reduced due to data incorporated as objects within objects. Switching screens will require very little computational time due to function calls and thus will occur very quickly. Updates occur in few seconds as data object is quickly circulated between classes using functions and can be accessed in minimal time complexity due to efficient STL data structures.

2.1.2.2 Safety Requirements

The software will not affect the data stored outside its database nor will it affect any other databases installed on the PC. It cannot cause any damage to the PC or its internal components.

2.1.2.3 Security Requirements

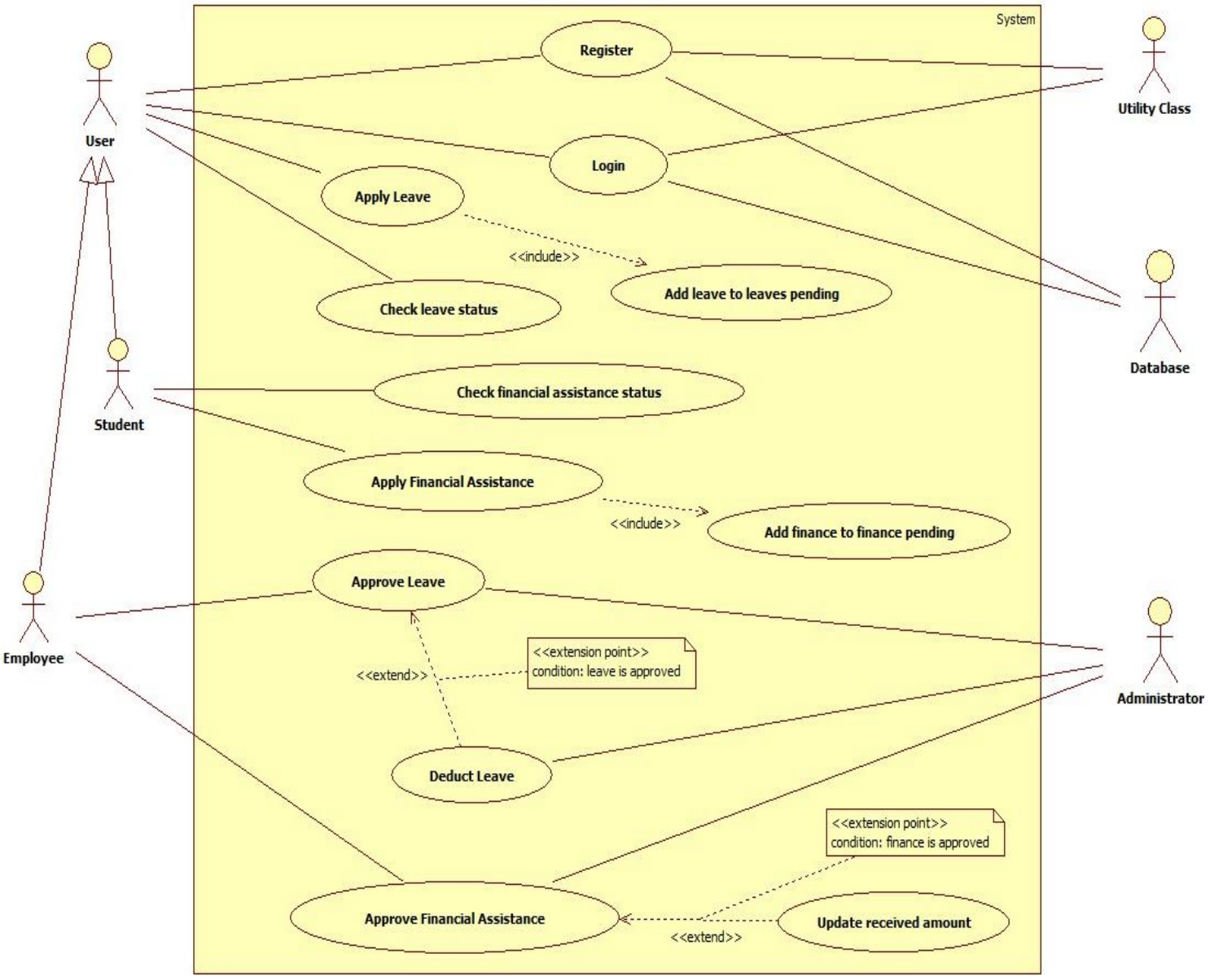
Every user has to register himself into the database before he can use all aspects of the software. As a part of the registration process he will be required to set a unique user identification number and a password, which would be used to grant him access whenever he wants to use the software. As a precautionary measure, to avoid any typological errors, the user would be asked to retype the password at the time of registration



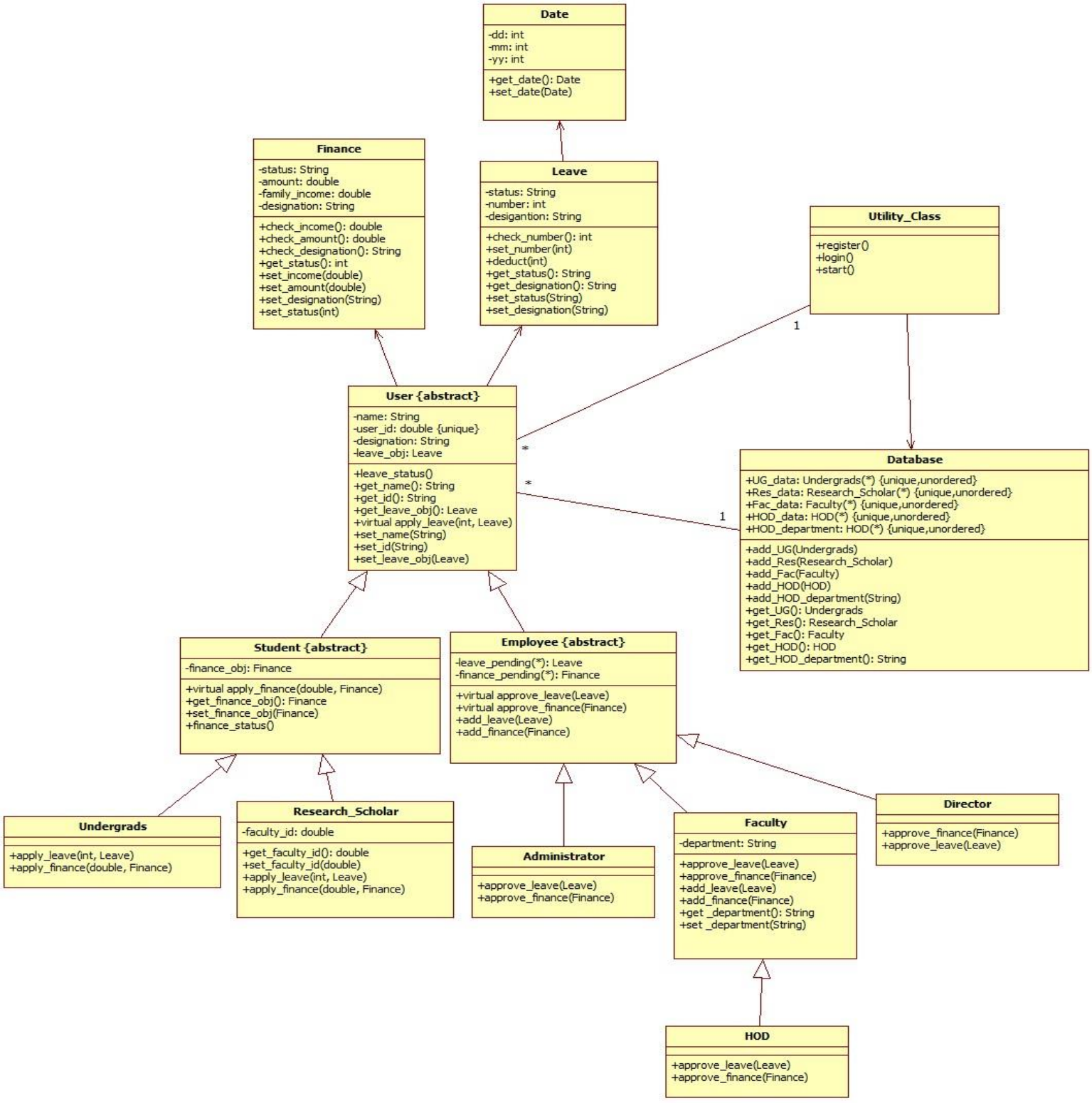
Activity diagram for user authentication

2.2 UML Diagrams

2.2.1 Use Case diagram

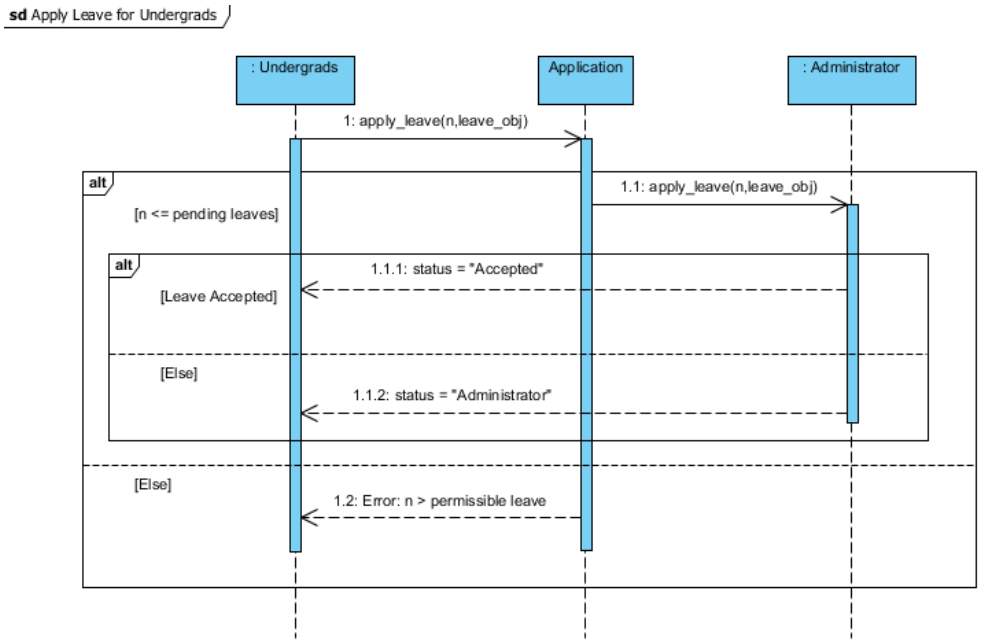


2.2.2 Class diagram

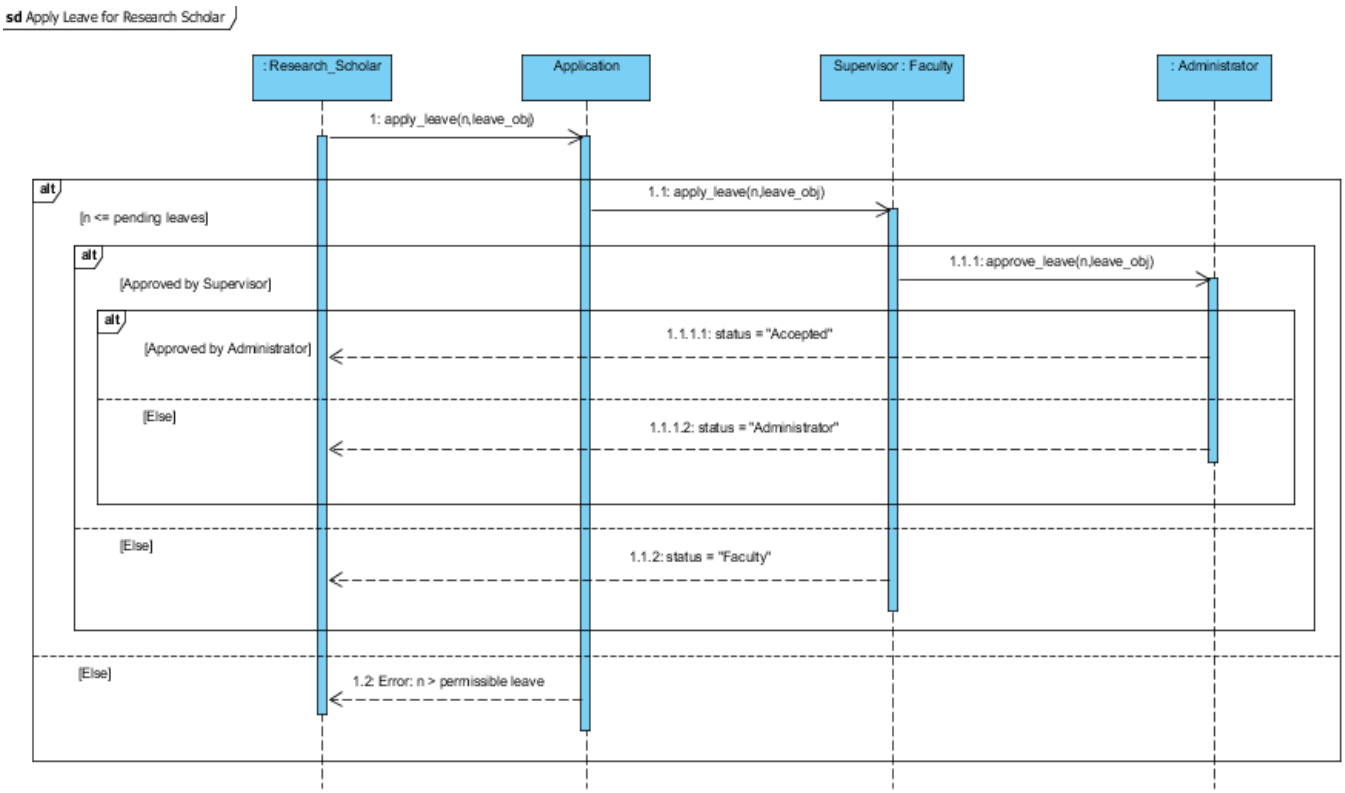


2.2.3 Sequence Diagrams

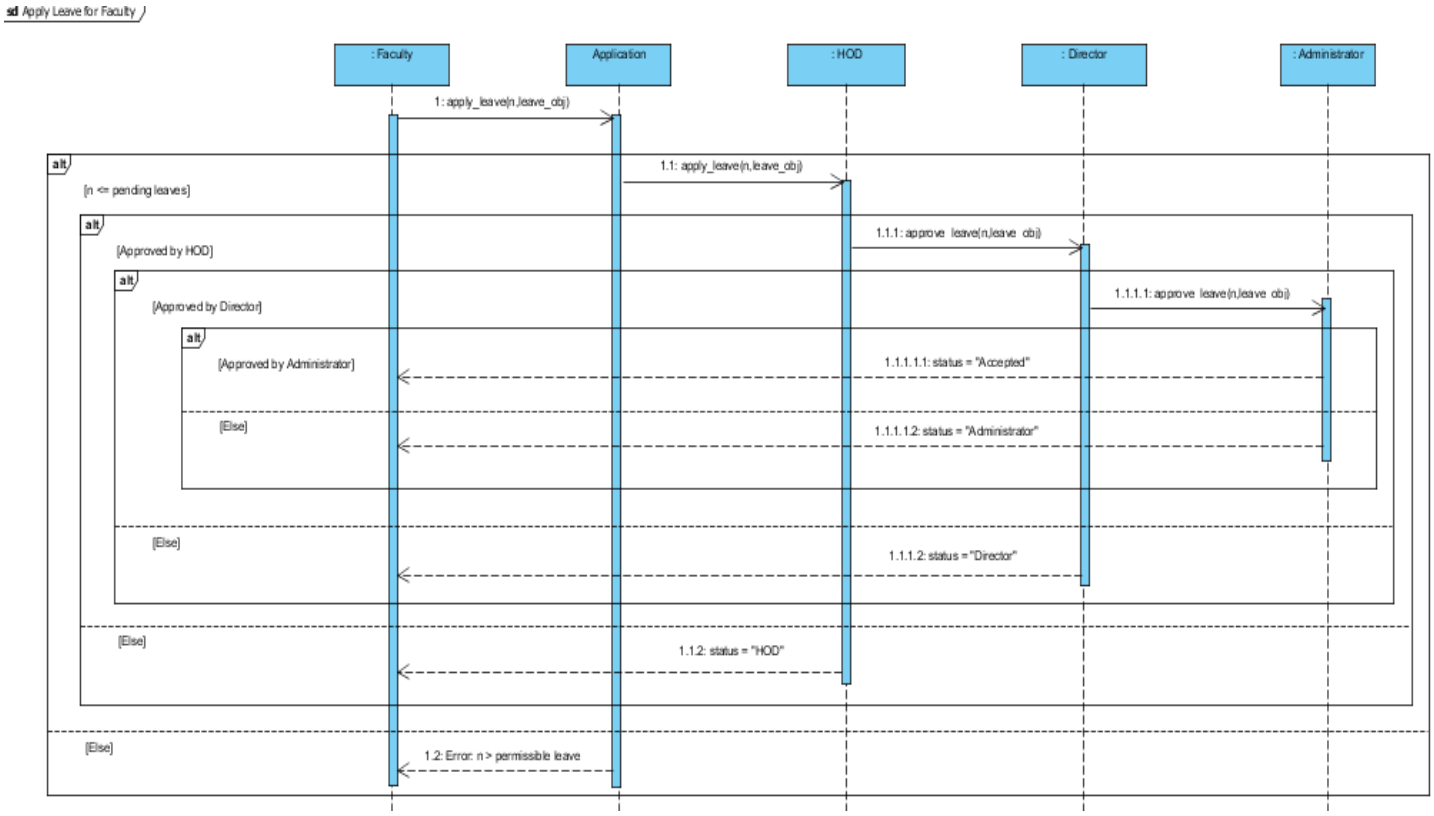
2.2.3.1 Leave Application - Undergrads



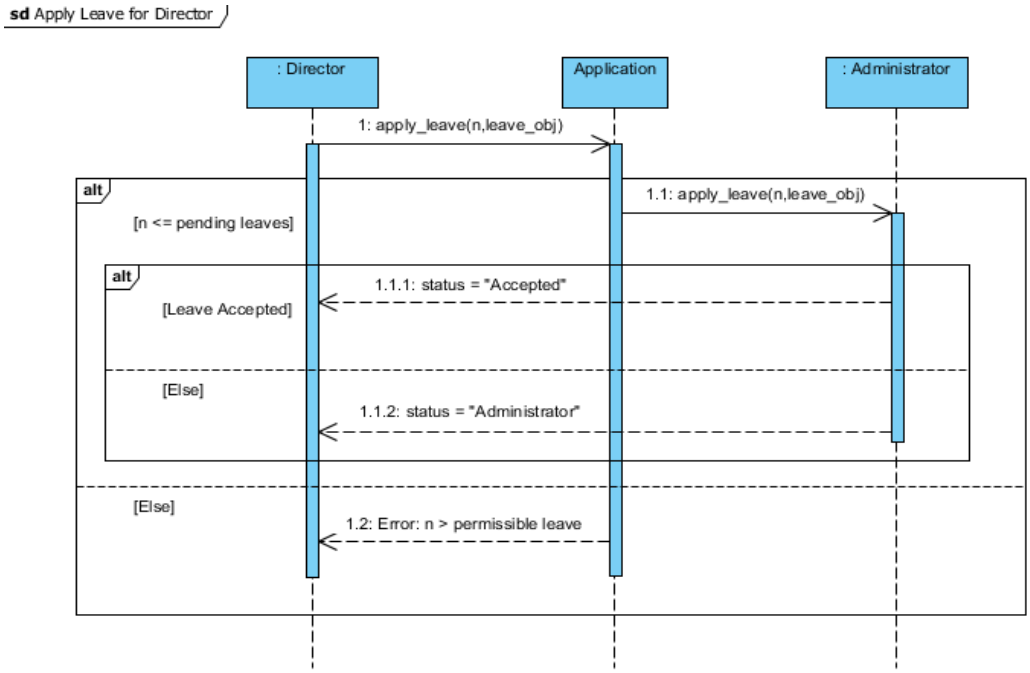
2.2.3.2 Leave Application - Research Scholars



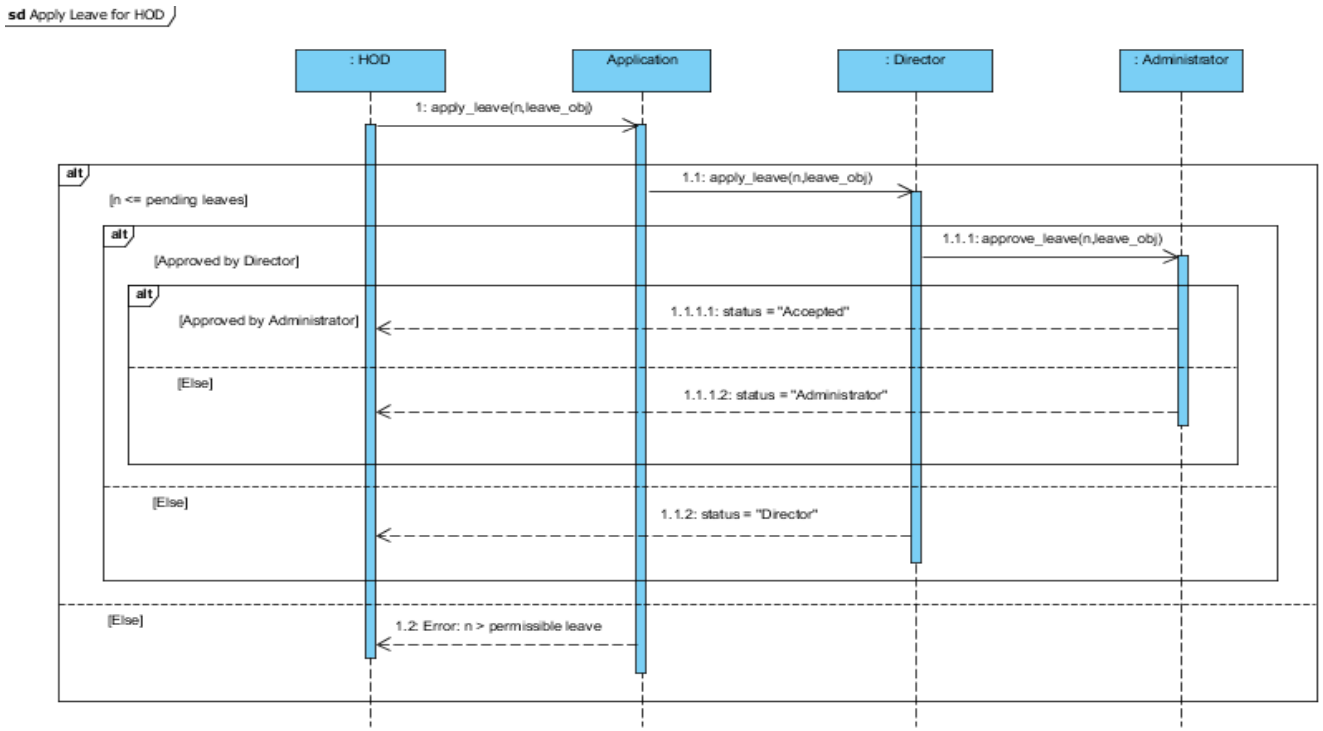
2.2.3.3 Leave Application – Faculty



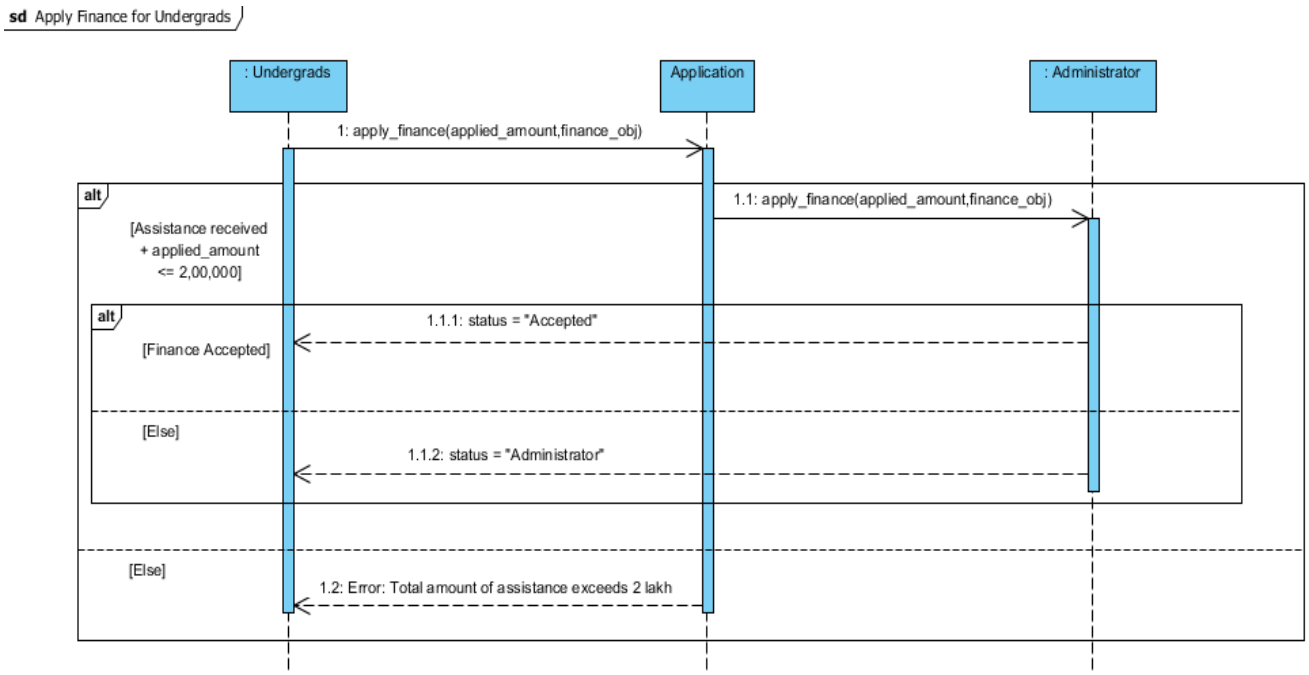
2.2.3.4 Leave Application – Director



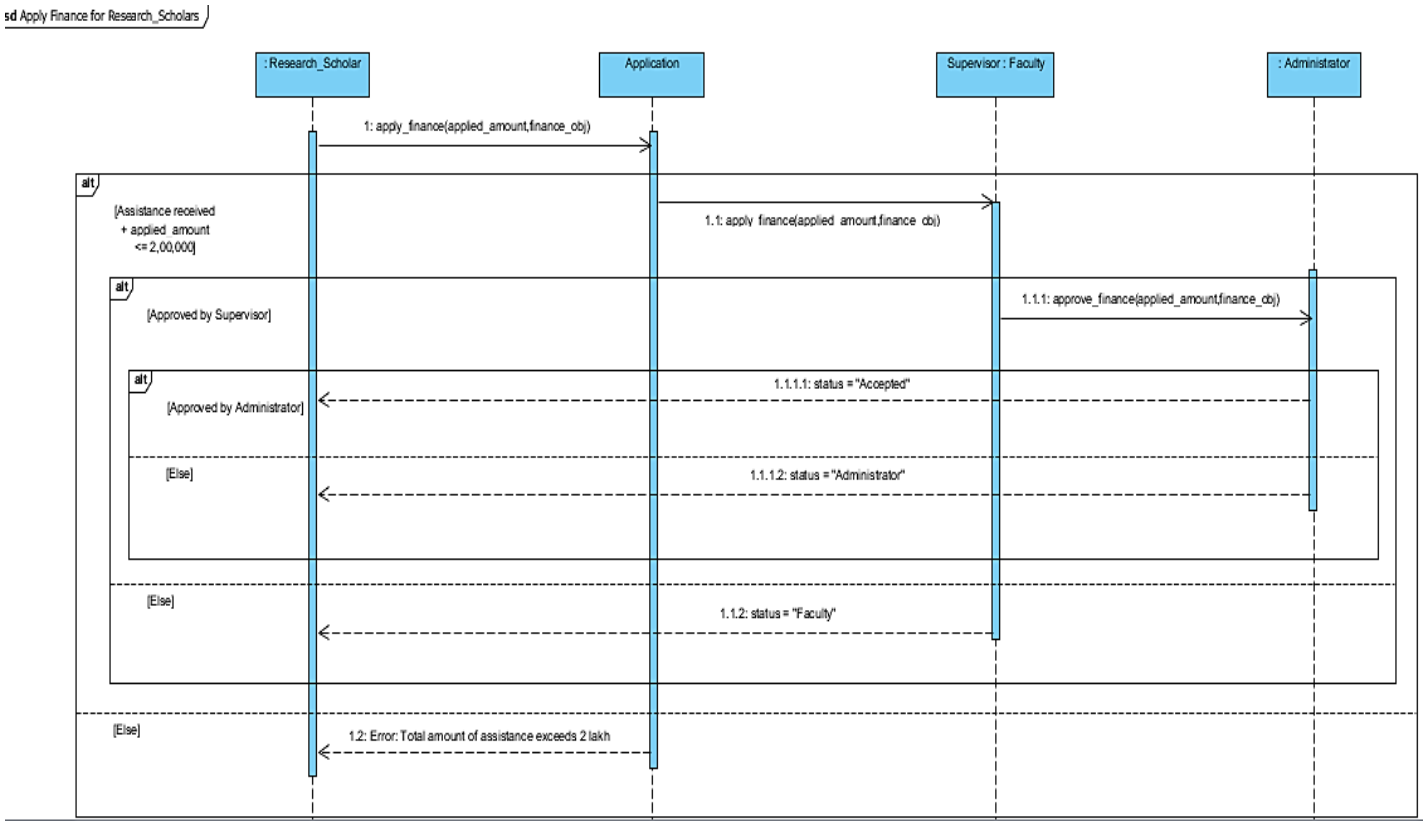
2.2.3.5 Leave Application – HOD



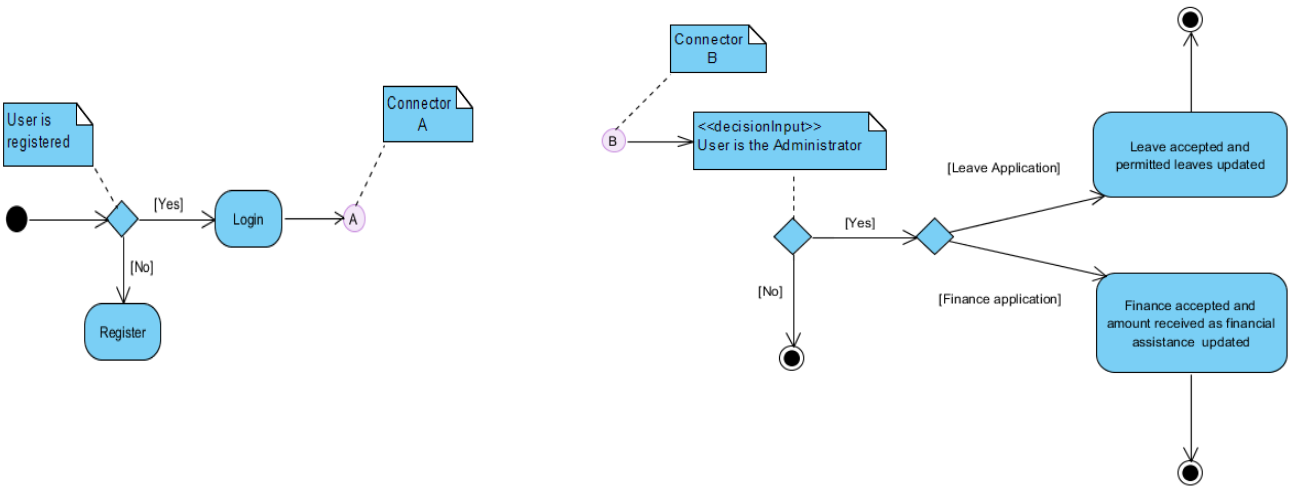
2.2.3.6 Financial assistance application – Undergrads

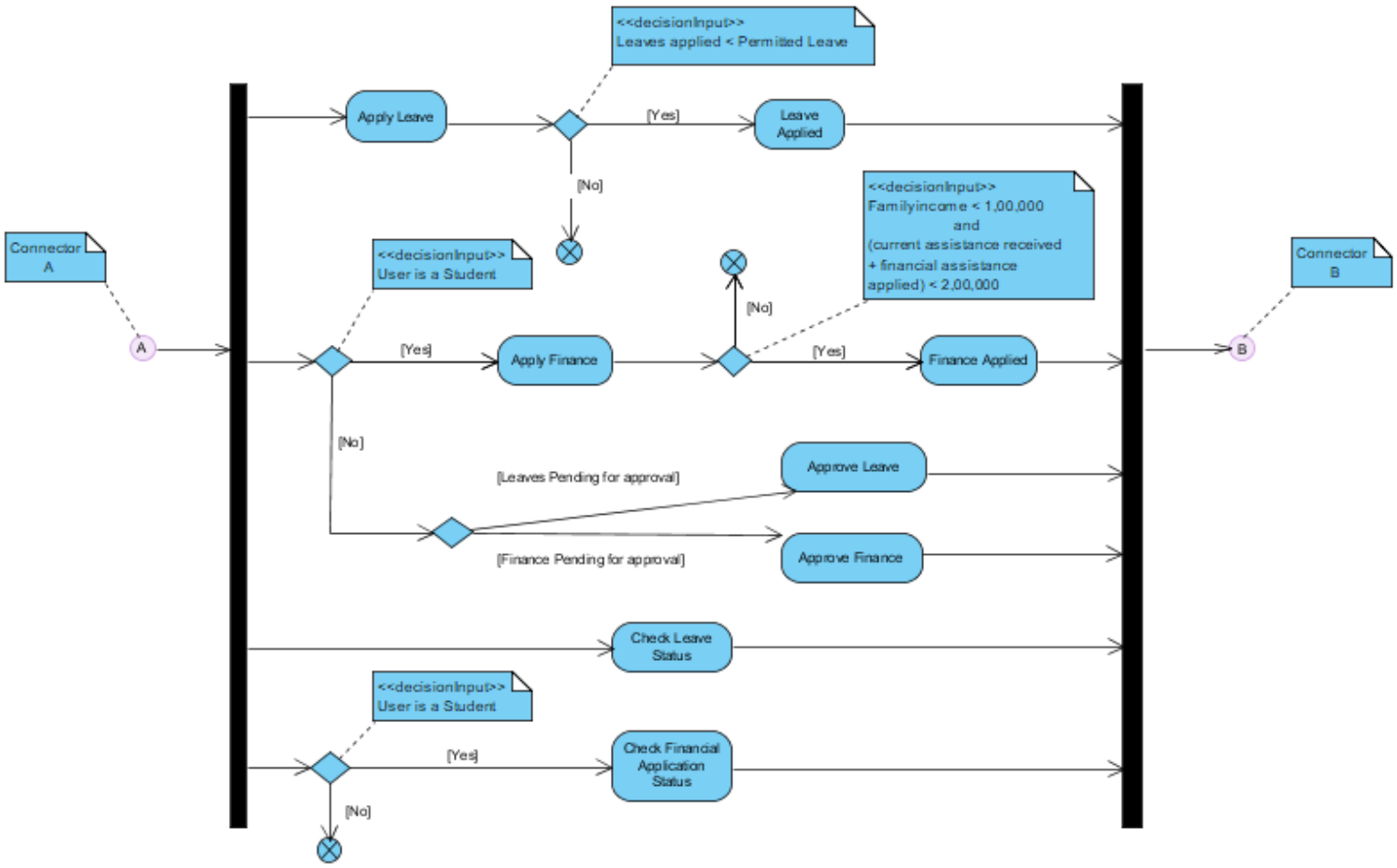


2.2.3.7 Financial Assistance Application – Research_Scholars



2.2.4 Activity Diagram





2.3 User Characteristics

1. There are two broad categories of the user interacting with the system: Student and Employee.
2. The student category encompasses Undergrads and Research scholar.
3. The employee category includes Faculty, Director and an Administrator. HOD is a Faculty but performs a separate role based on his authority.
4. The undergrad can apply for a Leave and apply for Financial assistance which is approved by higher authority. He can check his application status to make sure till where has his leave/financial assistance assistance processed or approved.
5. The research scholar performs the same functions as the student but the flow of leave/financial assistance application is modeled differently.
6. Each of the employee has the authority to approve a leave or financial assistance application and process it to the next higher authority. They themselves can apply for a leave application but not for financial assistance.

7. The administrator is supreme approver of any application in this hierarchy chain of workflow in the institute. He approves any application and has the power to deduct corresponding leaves from the remaining leave quota of the applicant after approval.

3. Specific Requirements

3.1 Use case description

Use Case Name:	Register
Actor:	User
Description:	Every new user must register before gaining access to the system. A user object based on user details is created and is added to the corresponding data structure which is maintaining the data organization in the software.
Preconditions:	The user must not be already registered.
Postconditions:	A unique user registration object is created which facilitates the user to interact with the system.
Priority:	No priorities are required for registering.
Frequency of Use:	A registration is done every time a new user accesses the system.
Normal Course of Events:	The user is asked to enter his name, designation and an id. The id is unique and is based on user's choice for his/her unique identification. These details are user specific and are later used for login.
Alternative Courses:	None
Exceptions:	A user id which the user tries to use may be earlier chosen.So, in that case the user should enter a different id.
Includes:	A call to STL functions to add the user object in corresponding data structure.
Special Requirements:	None
Assumptions:	A utility class handling the system control has the permission to register new users.
Notes and Issues:	There are different kinds of users. A separate STL data structure for each user is maintaining the track of their details. The utility class is not modeled separately in use cases but performs the specific system related non functional requirements.

Use Case Name:	Apply leave
Actor:	Undergrad
Description:	Every undergrad applies for a leave using this use case. The number of leaves applied for and the leave details of the applicant are passed to the next higher authority. Here the next authority is Administrator (office of academics). The leave application is added to administrator's record of pending leave applications which he can approve or not based on his wish.
Preconditions:	The undergrad can apply for leave iff no. of applied leaves is less than the no. of permitted leaves remaining in his/her leave quota.
Postconditions:	Once the leave is applied. The leave is added to pending list of leaves to be approved in administrator's account. Also the leave status of the undergrad now shows : administrator as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for leaves using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time a undergrad wants to apply for a leave.
Normal Course of Events:	The undergrad is asked to enter the number of leaves he wants to apply. If the leaves asked for are less than the permitted leaves the leave is applied and is forwarded to administrator for approval.
Alternative Courses:	None
Exceptions:	A leave application may not occur if leaves applied for are greater than the leaves permitted in which a notification of rejection may be generated on the screen.
Includes:	A call to STL functions to add the leave object in corresponding data structure.
Special Requirements:	None
Assumptions:	None
Notes and Issues:	There are different kinds of users. Each user's leave application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	Apply leave
Actor:	Research scholar
Description:	Every research scholar applies for a leave using this use case. The number of leaves applied for and the leave details of the applicant are passed to the next higher authority. Here the next authority is his/her faculty advisor. The leave application is added to faculty advisor's record of pending leave applications which he can approve or not based on his wish.
Preconditions:	The research scholar can apply for leave if f no. of applied leaves is less than the no. of permitted leaves remaining in his/her leave quota.
Postconditions:	Once the leave is applied. The leave is added to pending list of leaves to be approved in faculty advisor's account. Also the leave status of the research scholar now shows: faculty advisor as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for leaves using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time a research scholar wants to apply for a leave.
Normal Course of Events:	The research scholar is asked to enter the number of leaves he wants to apply. If the leaves asked for are less than the permitted leaves the leave is applied and is forwarded to faculty advisor for approval.
Alternative Courses:	None
Exceptions:	A leave application may not occur if leaves applied for are greater than the leaves permitted in which a notification of rejection may be generated on the screen.
Includes:	A call to STL functions to add the leave object in corresponding data structure.
Special Requirements:	None
Assumptions:	None
Notes and Issues:	There are different kinds of users. Each user's leave application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	Apply leave
Actor:	Faculty advisor
Description:	Every faculty advisor applies for a leave using this use case. The number of leaves applied for and the leave details of the applicant are passed to the next higher authority. Here the next authority is HOD of the corresponding department. The leave application is added to HOD's record of pending leave applications which he can approve or not based on his wish.
Preconditions:	The faculty can apply for leave iff no. of applied leaves is less than the no. of permitted leaves remaining in his/her leave quota.
Postconditions:	Once the leave is applied. The leave is added to pending list of leaves to be approved in HOD's account. Also the leave status of the faculty now shows: HOD as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for leaves using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time a faculty advisor wants to apply for a leave.
Normal Course of Events:	The faculty advisor is asked to enter the number of leaves he wants to apply. If the leaves asked for are less than the permitted leaves the leave is applied and is forwarded to HOD for approval.
Alternative Courses:	None
Exceptions:	A leave application may not occur if leaves applied for are greater than the leaves permitted in which a notification of rejection may be generated on the screen.
Includes:	A call to STL functions to add the leave object in corresponding data structure.
Special Requirements:	None
Assumptions:	The leave of a faculty advisor is forwarded only to the HOD of the corresponding department. Due to which we create a separate data structure in our utility class where departments are matched to the corresponding HOD for the ease of access. Note that number of HODs in the department are limited, so the space complexity is not increased much
Notes and Issues:	There are different kinds of users. Each user's leave application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	Apply leave
Actor:	HOD
Description:	Every HOD applies for a leave using this use case. The number of leaves applied for and the leave details of the applicant are passed to the next higher authority. Here the next authority is Director. The leave application is added to director's record of pending leave applications which he can approve or not based on his wish.
Preconditions:	The HOD can apply for leave iff no. of applied leaves is less than the no. of permitted leaves remaining in his/her leave quota.
Postconditions:	Once the leave is applied. The leave is added to pending list of leaves to be approved in director's account. Also the leave status of the undergrad now shows : director as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for leaves using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time a HOD wants to apply for a leave.
Normal Course of Events:	The HOD is asked to enter the number of leaves he wants to apply. If the leaves asked for are less than the permitted leaves the leave is applied and is forwarded to director for approval.
Alternative Courses:	None
Exceptions:	A leave application may not occur if leaves applied for are greater than the leaves permitted in which a notification of rejection may be generated on the screen.
Includes:	A call to STL functions to add the leave object in corresponding data structure.
Special Requirements:	None
Assumptions:	None
Notes and Issues:	There are different kinds of users. Each user's leave application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	Apply leave
Actor:	Director
Description:	The director applies for a leave using this use case. The number of leaves applied for and the leave details of the applicant are passed to the next higher authority. Here the next authority is Administrator (office of academics). The
	leave application is added to administrator's record of pending leave applications which he can approve or not based on his wish.
Preconditions:	The director can apply for leave iff no. of applied leaves is less than the no. of permitted leaves remaining in his/her leave quota.
Postconditions:	Once the leave is applied. The leave is added to pending list of leaves to be approved in administrator's account. Also the leave status of the undergrad now shows : administrator as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for leaves using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time the director wants to apply for a leave.
Normal Course of Events:	The director is asked to enter the number of leaves he wants to apply. If the leaves asked for are less than the permitted leaves the leave is applied and is forwarded to administrator for approval.
Alternative Courses:	None
Exceptions:	A leave application may not occur if leaves applied for are greater than the leaves permitted in which a notification of rejection may be generated on the screen.
Includes:	A call to STL functions to add the leave object in corresponding data structure.
Special Requirements:	None
Assumptions:	The director is the highest authority who can apply for an application.SO, this is the last possible implementation of our virtual use case: apply leave.
Notes and Issues:	There are different kinds of users. Each user's leave application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	Apply finance
Actor:	Undergrad
Description:	Every undergrad applies for a financial application using this use case. The amount applied for and the financial status details of the applicant are passed to the next higher authority. Here the next authority is Administrator (office of academics). The financial assistance application is added to administrator's record of pending financial applications which he can approve or not based on his wish.
Preconditions:	The undergrad can apply for financial iff his family income is below a certain criteria. Also there is a maximum limit on the amount of scholarship which can be obtained. So, the financial assistance which he/she is receiving must not cross that upper limit. So, these two preconditions need to be satisfied for the application to proceed.
Postconditions:	Once the financial assistance application is applied. The application is added to pending list of financial applications to be approved in administrator's account. Also the financial application status of the undergrad now shows: administrator as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for financial assistance applications using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time an undergrad wants to apply for a financial assistance.
Normal Course of Events:	The undergrad is asked to enter the amount for financial assistance he wants to apply for. If the preconditions are satisfied the financial assistance is applied and is forwarded to administrator for approval.
Alternative Courses:	None
Exceptions:	A financial assistance may not be applied if the preconditions are not satisfied in which a notification will be generated on the screen..
Includes:	A call to STL functions to add the finance object in corresponding data structure.
Special Requirements:	None
Assumptions:	None

Notes and Issues:	There are different kinds of users. Each user's financial application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	**Apply finance**
Actor:	Research scholar
Description:	Every research scholar applies for a financial application using this use case. The amount applied for and the financial status details of the applicant are passed to the next higher authority. Here the next authority is his/her faculty advisor . The financial assistance application is added to faculty advisor's record of pending financial applications which he can approve or not based on his wish.
Preconditions:	The research scholar can apply for financial iff his family income is below a certain criteria. Also there is a maximum limit on the amount of scholarship which can be obtained. So, the financial assistance which he/she is receiving must not cross that upper limit. So, these two preconditions need to be satisfied for the application to proceed.
Postconditions:	Once the financial assistance application is applied. The application is added to pending list of financial applications to be approved in faculty advisor's account. Also the financial application status of the research scholar now shows: faculty advisor as the status.
Priority:	Whenever a new application is filed, it is placed below the already applied for financial assistance applications using a queue data structure to maintain proper ordering of application.
Frequency of Use:	The use case is used every time a research scholar wants to apply for a financial assistance.
Normal Course of Events:	The research scholar is asked to enter the amount for financial assistance he wants to apply for. If the preconditions are satisfied the financial assistance is applied and is forwarded to faculty advisor for approval.
Alternative Courses:	None

Exceptions:	A financial assistance may not be applied if the preconditions are not satisfied in which a notification will be generated on the screen..
Includes:	A call to STL functions to add the finance object in corresponding data structure.
Special Requirements:	None
Assumptions:	None
Notes and Issues:	There are different kinds of users. Each user's financial application follows different chain of authority during it's approval based on the domain constraints of workflow management in the academic institute..

Use Case Name:	**Check leave status**
Actor:	Undergrad, research scholar, faculty advisor, HOD, Director
Description:	Every actor who has applied for a leave application can check the status to make sure till where has his application been processed in the given workflow.
Preconditions:	One who is checking the status must have applied for the leave.
Postconditions:	The status is notified to the user on the screen.
Priority:	None.
Frequency of Use:	As and when a user who has applied for a leave wants to check the status.
Normal Course of Events:	Every user who is eligible to apply for a leave has a leave object having the corresponding leave details. Any attribute of leave is the leave status of the user. The status is fetched whenever a request to check it is issued..
Alternative Courses:	None
Exceptions:	A user who has not applied for a leave may try to check the status under which nothing is displayed to him.
Includes:	A call to a getter function of the leave object to fetch the leave status.
Special Requirements:	None
Assumptions:	The user has applied for the leave.
Notes and Issues:	Every user can apply for a leave and hence check it's status accept the administrator as it is a passive actor and has no active role.

Use Case Name:	Check financial status
Actor:	Undergrad, research scholar
Description:	Every actor who has applied for a financial assistance application can check the status to make sure till where has his application been processed in the given workflow.
Preconditions:	One who is checking the status must have applied for the financial assistance.
Postconditions:	The status is notified to the user on the screen.
Priority:	None.
Frequency of Use:	As and when a user who has applied for a financial assistance wants to check the status.
Normal Course of Events:	Every user who is eligible to apply for a financial assistance has a finance object having the corresponding finance details. Any attribute of finance is the finance status of the user. The status is fetched whenever a request to check it is issued..
Alternative Courses:	None
Exceptions:	A user who has not applied for a financial assistance may try to check the status under which nothing is displayed to him.
Includes:	A call to a getter function of the finance object to fetch the leave status.
Special Requirements:	None
Assumptions:	The user has applied for the financial assistance.
Notes and Issues:	Only a student can apply for a financial assistance and hence he/she only can check the financial assistance application status.

Use Case Name:	Approve leave
Actor:	faculty advisor
Description:	A faculty advisor has a list of pending leaves under him which need to be approved.
Preconditions:	There must be pending leaves under him.
Postconditions:	After he approves the leave it goes to the administrator for approval and the status is changed accordingly.
Priority:	None.

Frequency of Use:	Whenever the faculty advisor wishes to approve the pending leaves under him he accesses the use case.
Normal Course of Events:	Once a research scholar has applied for the leave it goes to the corresponding faculty advisor for approval. Once he/she approves the leave it goes to the office of academics for approval.
Alternative Courses:	None
Exceptions:	There might be case where even though the leaves are pending but the faculty advisor may not approve them because he has the right to exercise his discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved leave from the data structure and the call to corresponding setter function to change the leave status accordingly.
Special Requirements:	None
Assumptions:	The research scholar has already applied for some leave, then only the leave can reach faculty advisor for approval.
Notes and Issues:	Only an employee or the administrator has the right to approve the leaves which are pending under them. But since the authorization structure is different we model the approve leave use case separately for all of them based on the workflow defined in the assumption..

Use Case Name:	**Approve leave**
Actor:	HOD
Description:	A HOD has a list of pending leaves under him which need to be approved.
Preconditions:	There must be pending leaves under him.
Postconditions:	After he approves the leave it goes to the director for approval and the status is changed accordingly.
Priority:	None.
Frequency of Use:	Whenever the HOD wishes to approve the pending leaves under him he accesses the use case.
Normal Course of Events:	Once a faculty advisor has applied for the leave it goes to the corresponding HOD for approval. Once he/she approves the leave it goes to the director for approval.
Alternative Courses:	None

Exceptions:	There might be case where even though the leaves are pending but the HOD may not approve them because he has the right to exercise his discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved leave from the data structure and the call to corresponding setter function to change the leave status accordingly.
Special Requirements:	None
Assumptions:	The faculty advisor has already applied for some leave, then only the leave can reach HOD for approval.
Notes and Issues:	Only an employee or the administrator has the right to approve the leaves which are pending under them. But since the authorization structure is different we model the approve leave use case separately for all of them based on the workflow defined in the assumption..

Use Case Name:	**Approve leave**
Actor:	Director
Description:	The director has a list of pending leaves under him which need to be approved.
Preconditions:	There must be pending leaves under him.
Postconditions:	After he approves the leave it goes to the administrator for approval and the status is changed accordingly.
Priority:	None.
Frequency of Use:	Whenever the director wishes to approve the pending leaves under him he accesses the use case.
Normal Course of Events:	Once a HOD has applied for the leave it goes to the director for approval. Once he/she approves the leave it goes to the office of academics for approval.
Alternative Courses:	None
Exceptions:	There might be case where even though the leaves are pending but the director may not approve them because he has the right to exercise his discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved leave from the data structure and the call to corresponding setter function to change the leave status accordingly.

Special Requirements:	None
Assumptions:	The HOD has already applied or approved for some leave, then only the leave can reach director for approval.
Notes and Issues:	Only an employee or the administrator has the right to approve the leaves which are pending under them. But since the authorization structure is different we model the approve leave use case separately for all of them based on the workflow defined in the assumption..

Use Case Name:	**Approve leave**
Actor:	Administrator
Description:	The administrator has a list of pending leaves under him which need to be approved.
Preconditions:	There must be pending leaves under him.
Postconditions:	After he approves the leave the leave is finally accepted. The accepted leaves are deducted from the permitted leaves of the applicant and the status is finally changed to approved.
Priority:	None.
Frequency of Use:	Whenever the administrator wishes to approve the pending leaves under him he accesses the use case.
Normal Course of Events:	A leave for approval reaches the administrator irrespective of the applicant because it is the highest authority for approval and any leave filed must be approved by the administrator after which it is finally accepted.
Alternative Courses:	None
Exceptions:	There might be case where even though the leaves are pending but the administrator may not approve them because he has the right to exercise his discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved leave from the data structure and the call to corresponding setter function to change the leave status accordingly. There is also a call to deduct leave which makes sure that the maximum permitted leaves of the applicant are reduced accordingly.
Special Requirements:	None
Assumptions:	None.

Notes and Issues:	Only an employee or the administrator has the right to approve the leaves which are pending under them. But since
	the authorization structure is different we model the approve leave use case separately for all of them based on the workflow defined in the assumption..

Use Case Name:	**Approve finance**
Actor:	faculty advisor
Description:	A faculty advisor has a list of pending financial assistance applications under him which need to be approved.
Preconditions:	There must be pending financial applications under him.
Postconditions:	After he approves the financial application it goes to the administrator for approval and the status is changed accordingly.
Priority:	None.
Frequency of Use:	Whenever the faculty advisor wishes to approve the pending financial application under him he accesses the use case.
Normal Course of Events:	Once a research scholar has applied for the financial assistance it goes to the corresponding faculty advisor for approval. Once he/she approves the leave it goes to the office of academics for approval.
Alternative Courses:	None
Exceptions:	There might be case where even though the applications are pending but the faculty advisor may not approve them because he has the right to exercise his discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved financial application from the data structure and the call to corresponding setter function to change the finance status accordingly.
Special Requirements:	None
Assumptions:	The research scholar has already applied for some financial assistance, then only the application can reach faculty advisor for approval.

Notes and Issues:	Only an employee or the administrator has the right to approve the financial application which are pending under them. But since the authorization structure is different we model the approve finance use case separately for all of them based on the workflow defined in the assumption..

Use Case Name:	**Approve finance**
Actor:	HOD
Description:	A HOD has a list of pending financial assistance under him which need to be approved.
Preconditions:	There must be pending financial applications under him.
Postconditions:	After he approves the application it goes to the director for approval and the finance status is changed accordingly.
Priority:	None.
Frequency of Use:	Whenever the HOD wishes to approve the pending financial assistances under him he accesses the use case.
Normal Course of Events:	Once a faculty advisor has applied for the financial assistance it goes to the corresponding HOD for approval. Once he/she approves the financial assistance it goes to the director for approval.
Alternative Courses:	None
Exceptions:	There might be case where even though the financial applications are pending but the HOD may not approve them because he has the right to exercise his discretion. In that case the application will remain pending under him.
Includes:	A call to STL library functions to remove an approved financial application from the data structure and the call to corresponding setter function to change the finance status accordingly.
Special Requirements:	None
Assumptions:	The faculty advisor has already approved for some financial assistance, then only the financial assistance application can reach HOD for approval.
Notes and Issues:	Only an employee or the administrator has the right to approve the financial assistance which are pending under them. But since the authorization structure is different we

	model the approve finance use case separately for all of them based on the workflow defined in the assumption.

Use Case Name:	**Approve finance**
Actor:	Director
Description:	The director has a list of pending financial assistance applications under him which need to be approved.
Preconditions:	There must be pending financial applications under him.
Postconditions:	After he approves the financial assistance application it goes to the administrator for approval and the status is changed accordingly.
Priority:	None.
Frequency of Use:	Whenever the director wishes to approve the pending financial applications under him he accesses the use case.
Normal Course of Events:	Once a HOD has applied for the financial assistance it goes to the director for approval. Once he/she approves the application it goes to the office of academics for approval.
Alternative Courses:	None
Exceptions:	There might be case where even though the financial applications are pending but the director may not approve them because he has the right to exercise his discretion. In that case the application will remain pending under him.
Includes:	A call to STL library functions to remove an approved finance from the data structure and the call to corresponding setter function to change the finance status accordingly.
Special Requirements:	None
Assumptions:	The HOD has already approved for some finance application, then only the finance application can reach director for approval.
Notes and Issues:	Only an employee or the administrator has the right to approve the financial assistance which are pending under them. But since the authorization structure is different we model the approve finance use case separately for all of them based on the workflow defined in the assumption

Use Case Name:	Approve finance
Actor:	Administrator
Description:	The administrator has a list of pending financial applications under him which need to be approved.
Preconditions:	There must be pending financial assistance applications under him.
Postconditions:	After he approves the application, financial assistance is finally granted. The financial assistance which the student is currently withdrawing is added and adjusted accordingly.
Priority:	None.
Frequency of Use:	Whenever the administrator wishes to approve the pending financial application under him he accesses the use case.
Normal Course of Events:	A financial assistance application for approval reaches the administrator irrespective of the applicant because it is the highest authority for approval and any application filed must be approved by the administrator after which it is finally accepted.
Alternative Courses:	None
Exceptions:	There might be case where even though the financial applications are pending but the administrator may not approve them because he has the right to exercise his
	discretion. In that case the leave will remain pending under him.
Includes:	A call to STL library functions to remove an approved finance from the data structure and the call to corresponding setter function to change the finance status accordingly. There is also a call to add amount which makes sure that the maximum permitted amout of the applicant is more than the assistance being currently provided to him,
Special Requirements:	None
Assumptions:	Only a student can apply for financial assistance scholarship.
Notes and Issues:	Only an employee or the administrator has the right to approve the financial assistance which are pending under them. But since the authorization structure is different we model the approve finance use case separately for all of them based on the workflow defined in the assumption

3.2 Reliability

3.2.1 Maintenance

For applications to be approved by the higher authorities a queue is maintained so that the one who applies first gets preference during approval. It allows easy maintenance.
Forwarding or migration of application without approval from one authority to another is not supported to ensure that application status is not corrupted. It also prevents code fragmentation.

3.2.2 Maximum bug rate

Efforts are made to minimize any incorporation of bug into the product. However, there may be 1 major bug in thousand lines of code. Minor bugs are also carefully rectified. There might be 2-3 minor bugs in thousand lines of code. There will be a maximum of 1 bug (major)/ KLOC.

3.2.3 Security Considerations

The Workflow Management system is particular in ensuring the safety and security of its users. The user data is not shared with anybody and is safely encapsulated in the data structures in software. Also only the authenticated users can access the software after prior registration and proper login which is equipped with proper password implementation techniques.

3.3 Performance Requirements

3.3.1 Response time

The maximum response time for the application of a leave or financial assistance shall be within 10 seconds based on the processing. The status checking involves a searching procedure, efficient searching shall keep the response time within 30 seconds of processing.
However, the approval is subject to the higher authority and may be time indefinite based on when he/she is willing to approve. But once approved the processing shall finish within 30 seconds of commencement.

3.3.2 Capacity

The user can apply for leave or financial assistance only if he/she is not having any application pending/under processing. The maximum number of applications to be approved is limited only by the capacity of the authority approving them; there is no upper limit inherent in the Workflow

management for leave/financial assistance application as many users may be applying and approving them simultaneously.

3.3.3 Deadline sensitivity

Assuming applications applied and approved are accurate, the Workflow Management will ensure that all applications are accepted within 5% error allowance based on the process constraints.

3.4 Supportability

3.4.1 Naming Convention

All code will be written as specified by the underscore_case. The Hungarian naming convention has gone relatively out of date.

3.4.2 Coding Standards

1. Limiting the use of global: Efforts will be made to remove the global attributes and functions to avoid data corruption and ensure security via data encapsulation.

2. Naming conventions: Underscore case convention is followed in naming attributes and functions for clarity. Efforts are made to ensure that the name conveys the partial meaning of the role to be performed by the attribute or function.

3. Well-documented: Proper comments will be written to ensure that separate pieces of code convey proper meaning and utility of their implementation.

4. Goto statements: To properly ensure minimization of errors and understandability of code, efforts will be made to eliminate goto statements.

5. Modular coding: The length of any function should not exceed 100-200 source lines. A function that is very lengthy is usually very difficult to understand as it probably carries out many different functions. Although, functions involving switch cases and branching may extend a bit longer to incorporate function logic.

6. Errors and Exception handling: All error conditions will be considered and flagged for the calling routine's attention.

3.5 Design Constraints

3.5.1 Software Language

All the coding will be done in standard C++ programming language.

3.5.2 As the software runs in a terminal window of a C++ IDE, concurrent updates cannot be permitted, thereby restricting two different users from working simultaneously.

3.6 Interfaces

3.6.1 User Interfaces

Any user of this application would be directed to the log-in page in the terminal window when he/she runs the application. If the user is new, he/she would be required to register before using the application. An existing user would be directed to his/her profile page after a successful login.
Every user should have a profile page where they can do one or more of the possible, depending on their designation:

- Apply for a leave
- Apply for financial assistance
- Approve a leave application
- Approve a financial assistance application

4. Supporting Information

APPENDIX A: GLOSSARY

Table-1

Term	Definition
Leave	The leave object keeps track of remaining leaves. It keeps the track of status while processing of leave applications. It includes the data of issue and ensures who has issued it.
Finance	A finance object is used for applying financial assistance application. It contains family income of applicant for comparison based approval and can be used to track status of approval like leave.
User	Any individual in the academic institute.
Student	One who studies in the academic institute
Employee	One who is employed in the academic institute
Undergrads	A category of student class
Research _Scholar	A category of student class studying under a faculty
Faculty	An employee teaching in the institute and assisting a research scholar
HOD	He is the head of the department and an employee in the institute
Director	He is an employee and is the head of the academic institute
Administrator	A passive actor. It refers to office of academics whose final approval is required for approving any application.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the various constraints under which it operates. For example, this document.
Stakeholder	Any person other than the developer, with an interest in the project.

Table-2

Function Name	Task Description
apply_leave	A user can apply for a leave
apply_finance	A user can apply for financial assistance
Approve_leave	An authority can approve a leave pending under him
Approve_finance	An authority can approve a financial assistance application pending under him

Leave_status	A user can check under which authority his leave application is pending or if it has been approved
Finance_status	A user can check under which authority his financial assistance application is pending or if it has been approved
deduct	It deducts the number of permitted leaves after approval of leave application
Add_amount	It adds the total amount which a student is withdrawing after his financial assistance is approved
Get_department	Returns the department of a faculty advisor
Get_faculty_id	It returns the id of a faculty advisor under which a research scholar is working
Other getter functions	For getting the attributes of classes
Other setter functions	For setting the attributes of the classes
Get_finance_obj	It fetches the financial details of the user
Get_leave_obj	It fetches the leave details of the user
Add functions	To add new users to database

APPENDIX B: Conditions And Actions (Tabular Analysis Of Functional Requirements)

Assumptions:

1. Maximum leaves permitted to every user is permitted leaves = 50.
2. Only a student with family income below 1,00,000 INR can seek financial assistance.
3. The maximum financial assistance which can be sought after is 2,00,000 INR.
4. Leave number(in leave object)= Permitted leaves – total leaves taken till now

Table-3

Condition	Action
Leave Number =0	No leave can be applied
Leave number >0 and applied leave number> leave number	No leave applied
Leave number >0 and applied leave number< Leave number	Leave can be applied and if leave is approved leave number=leave number-applied leaves
Family income>1,00,000 INR	No financial assistance application can be applied

Family income <1,00,000 and amount=2,00,000	No financial assistance can be applied
Family income< 1,00,000 and amount<2,00,000 and financial assistance applied+amount>2,00,000	No financial assistance can be applied
Family income< 1,00,000 and amount<2,00,000 and financial assistance applied+amount<2,00,000	Financial assistance application is applied and if it gets approved then, amount=amount+financial assistance applied

APPENDIX C: Project Proposal

Working Title: Workflow Management System

Motivation:

We are designing an application that facilitates the process of easy status checking in a complex workflow management in an academic institute.The Users can apply for financial assistance and leaves based on their designations and functional constraints. They can be easily approved due to smooth data flow and most importantly, it is very easy for the user to keep track of his/her application in complex working environment. This is relevant specifically in large academic institutes, ranging from branched schools to universities. Depending on the number of people and/or the number of applications involved, these situations can get very complicated.

Problem Statement:

Organising the workflow management in an academic institute which helps in:

1. Checking the status of a leave applied by some user.
2. Checking the status of a financial assistance application applied by some user.

The software will solve the problem by facilitating instant status checking for the user by implementing a transaction based application where information flows in the form of data object and the users can continuously access or update them.

A**dditional Implemented Features**:

1. The users can apply for leave/financial applications.
2. The concerned authorities can approve the applications.

We shall also add the following features later if the time permits:

1. Appropriate user data handling in an extensive database.
2. A detailed application management where an application can even withdraw the application in between of it's scheduled processing if he wishes and cancel it.

Objectives:

- Besides implementing the above features, there are a number of additional objectives we must accomplish as a team:
- Develop a timeline detailing each stage of development
- Establish well-defined roles for each member of the group
- Create detailed software specifications
- Ensure each member is keeping up the pace and completing work on time
- Create a fully-functional, bug-free application